MAT 602 - Final Project: Predicting Early Hospital Readmissions

Javier Rojas. jrojas@stu.edu

Due 3/6/17 by 6:00 PM EST

1. State the problem/area you will focus on for your final project.

- As a means to resolve hospital readmission rates, part of the Patient Protection and Affordable Care Act penalizes hospitals with excessive readmissions at 30 days through a program called the Hospital Readmission Reduction Program.
- As such, hospitals are scrambling to target the most at-risk patients and reduce their readmission rates in the most cost effective manner.
- In this work, I studied a dataset comprising of 100,000 total patients from 5 different hosptals. My problem comprises of a binary supervised learning task where based on the features of each patient, the goal is to predict where or not the patient was readmitted to the hospital.
- In addition, a search for the 3 most important features whose results dictate whether or not the patient will be reamitted to the hospital was implemented.

2. Dataset

- Here is the dataset that was studied for this work.

In [100]:
```
import pandas as pd
import numpy as np

# Loading the dataset.
df = pd.read_csv('C:\\Users\\javyr\\Documents\\GitHub\\r-server-hospital-length-of-stay\\Data\\LengthOfStay.csv')

# Creating a binary variable 'Readmission', which is constructed from the number of readmissions
# 'rcount' by assigning the variable a value of '-1' when the corresponding patient was not
# readmitted and '1' otherwise.
df['Readmission'] = np.where(df['rcount']=='0', -1,1)

# Display the dataset.
df
```

Out[100]:

	eid	vdate	rcount	gender	dialysisrenalendstage	asthma	irondef	pneum	substancedependence	psychologicaldisordermajor	...	bloodureanitro	creatinine	bmi	pulse	respiration	secondarydiagnosisnonicd9	discharged	facid	lengthofstay	Readmission
0	1	8/29/2012	0	F	0	0	0	0	0	0	...	12.0	1.390722	30.432418	96	6.5	4	9/1/2012	B	3	-1
1	2	5/26/2012	5+	F	0	0	0	0	0	0	...	8.0	0.943164	28.460516	61	6.5	1	6/2/2012	A	7	1
2	3	9/22/2012	1	F	0	0	0	0	0	0	...	12.0	1.065750	28.843812	64	6.5	2	9/25/2012	B	3	1
3	4	8/9/2012	0	F	0	0	0	0	0	0	...	12.0	0.906862	27.959007	76	6.5	1	8/10/2012	A	1	-1
4	5	12/20/2012	0	F	0	0	0	1	0	1	...	11.5	1.242854	30.258927	67	5.6	2	12/24/2012	E	4	-1
5	6	11/27/2012	3	M	0	0	0	0	0	0	...	11.0	0.884663	27.875940	83	6.1	1	12/3/2012	B	6	1
6	7	9/27/2012	4	F	0	0	0	0	0	0	...	6.0	0.923083	29.928817	68	6.5	4	10/3/2012	A	6	1
7	8	6/4/2012	0	F	0	0	0	0	0	1	...	11.0	0.768106	32.999171	63	6.0	3	6/7/2012	E	3	-1
8	9	7/5/2012	0	F	0	0	0	0	1	0	...	12.0	1.178786	28.802682	69	6.5	1	7/8/2012	E	3	-1
9	10	9/1/2012	0	F	0	0	0	0	0	0	...	10.0	1.585072	32.503727	65	6.5	0	9/3/2012	E	2	-1
10	11	5/11/2012	0	F	0	0	0	0	0	0	...	7.0	1.254258	30.325155	73	6.5	1	5/12/2012	B	1	-1
11	12	4/21/2012	4	M	0	0	0	0	0	0	...	16.0	1.264943	27.301720	63	5.7	1	4/28/2012	A	7	1
12	13	7/8/2012	0	F	0	1	0	0	0	0	...	17.0	0.981130	31.357435	82	5.9	1	7/11/2012	D	3	-1
13	14	12/5/2012	0	F	0	0	0	0	1	0	...	12.0	1.022660	29.692904	73	6.5	1	12/10/2012	E	5	-1
14	15	6/15/2012	1	F	0	0	0	0	0	0	...	12.0	1.050820	28.577796	69	6.5	1	6/17/2012	A	2	1
15	16	3/14/2012	1	F	0	0	0	0	0	0	...	16.0	1.239281	30.639405	68	6.5	7	3/18/2012	C	4	1
16	17	12/10/2012	3	F	0	0	0	0	0	0	...	10.0	1.015645	32.052032	66	5.5	0	12/16/2012	B	6	1
17	18	10/24/2012	0	M	0	0	0	0	0	0	...	9.0	1.188563	31.103821	72	6.9	6	10/25/2012	A	1	-1
18	19	7/10/2012	0	M	0	0	0	0	1	0	...	7.0	0.798395	28.981023	63	7.4	1	7/15/2012	E	5	-1
19	20	6/6/2012	0	M	0	0	0	0	0	0	...	11.0	1.051098	31.822687	72	6.5	1	6/8/2012	B	2	-1
20	21	7/31/2012	0	F	0	0	0	0	0	0	...	10.5	1.054753	30.770009	75	5.5	1	8/3/2012	A	3	-1
21	22	8/24/2012	0	M	0	0	0	0	0	0	...	12.0	1.332474	28.931121	77	6.5	1	8/25/2012	A	1	-1
22	23	12/26/2012	3	M	0	0	0	0	0	0	...	28.0	1.065005	28.341693	70	6.2	1	1/1/2013	C	6	1
23	24	3/10/2012	1	M	0	0	1	0	0	0	...	35.0	1.071094	29.704510	101	5.9	1	3/14/2012	C	4	1
24	25	6/14/2012	0	F	1	0	0	1	1	0	...	104.0	1.248269	25.917837	62	5.6	1	6/20/2012	E	6	-1
25	26	7/8/2012	1	M	0	0	0	0	0	1	...	35.0	0.967228	31.255578	81	5.9	1	7/12/2012	E	4	1
26	27	11/27/2012	2	F	0	0	0	0	0	1	...	7.0	0.913336	35.482148	88	7.7	0	12/4/2012	E	7	1
27	28	6/28/2012	3	F	0	0	0	0	0	0	...	12.0	1.035963	30.072107	65	6.5	1	7/3/2012	B	5	1
28	29	5/13/2012	4	F	0	0	0	0	0	0	...	6.0	1.301718	27.983707	75	5.8	1	5/19/2012	B	6	1
29	30	1/12/2012	0	F	0	0	0	0	0	0	...	12.0	1.073185	31.623678	66	5.7	2	1/14/2012	A	2	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99970	99971	6/30/2012	5+	M	0	0	0	0	0	1	...	6.0	1.235638	32.092100	74	6.5	1	7/8/2012	A	8	1
99971	99972	6/7/2012	0	M	0	0	0	0	0	1	...	12.0	1.112022	30.036177	80	6.5	1	6/10/2012	E	3	-1
99972	99973	5/30/2012	3	F	0	0	0	0	0	1	...	16.0	0.922509	26.033738	69	7.2	1	6/7/2012	E	8	1
99973	99974	9/1/2012	0	M	1	0	0	0	0	1	...	33.0	1.243078	31.380436	53	5.4	2	9/7/2012	E	6	-1
99974	99975	5/21/2012	0	F	0	0	0	0	0	0	...	7.0	1.361405	27.360694	71	6.5	1	5/23/2012	A	2	-1
99975	99976	7/26/2012	3	F	0	0	1	0	0	0	...	10.0	1.052441	30.994204	77	5.4	1	8/1/2012	B	6	1
99976	99977	4/11/2012	0	M	0	0	0	0	0	0	...	12.0	1.124496	29.308392	74	6.5	2	4/12/2012	B	1	-1
99977	99978	11/21/2012	1	F	0	0	0	0	0	1	...	15.0	1.092245	29.868996	81	5.5	1	11/25/2012	E	4	1
99978	99979	2/26/2012	0	M	0	0	0	0	0	0	...	12.0	1.163947	31.696450	76	6.5	1	2/27/2012	B	1	-1
99979	99980	2/17/2012	1	M	0	0	1	0	0	1	...	18.0	1.220202	31.216115	80	5.8	1	2/22/2012	E	5	1
99980	99981	8/21/2012	0	F	0	0	0	0	0	0	...	18.0	1.087694	28.470080	75	6.7	2	8/24/2012	C	3	-1
99981	99982	1/30/2012	0	F	1	0	0	0	0	0	...	83.0	1.393124	28.052928	66	6.3	3	2/5/2012	E	6	-1
99982	99983	6/3/2012	0	M	0	0	0	0	0	1	...	12.0	1.346726	29.802711	82	6.9	1	6/8/2012	E	5	-1
99983	99984	7/21/2012	0	M	0	0	0	0	0	0	...	7.0	1.575308	31.682100	86	7.0	4	7/22/2012	B	1	-1
99984	99985	6/19/2012	2	F	0	0	0	0	0	0	...	12.0	1.099844	30.924695	77	6.5	1	6/24/2012	A	5	1
99985	99986	11/28/2012	0	M	0	0	1	0	0	0	...	42.0	1.228341	29.929659	90	5.5	1	12/3/2012	C	5	-1
99986	99987	4/6/2012	2	F	0	0	0	0	0	0	...	14.0	0.935627	28.282345	66	6.3	4	4/10/2012	B	4	1
99987	99988	8/23/2012	4	F	0	0	0	0	1	0	...	7.0	1.247808	29.773143	76	6.3	3	8/30/2012	E	7	1
99988	99989	9/5/2012	2	M	1	0	1	0	0	0	...	29.0	0.655632	31.700961	59	5.6	2	9/13/2012	E	8	1
99989	99990	4/22/2012	0	M	0	0	0	0	0	1	...	12.0	1.293929	30.141954	82	7.6	1	4/26/2012	E	4	-1
99990	99991	12/15/2012	0	F	0	0	0	0	0	0	...	12.0	1.141481	29.834409	69	6.5	1	12/17/2012	B	2	-1
99991	99992	4/19/2012	1	F	0	0	0	0	1	0	...	15.0	1.303726	32.193258	80	6.5	1	4/23/2012	E	4	1
99992	99993	11/19/2012	0	M	0	1	0	0	0	0	...	12.0	0.977250	28.941686	86	6.5	2	11/22/2012	E	3	-1
99993	99994	12/6/2012	2	F	0	0	0	0	0	0	...	7.0	0.914914	32.952413	66	4.8	1	12/12/2012	B	6	1
99994	99995	1/27/2012	4	F	0	0	0	0	0	0	...	12.0	1.446806	33.761061	65	6.5	9	2/2/2012	A	6	1
99995	99996	1/28/2012	3	M	0	0	0	0	0	0	...	12.0	0.650323	30.063069	80	6.5	1	2/3/2012	B	6	1
99996	99997	8/6/2012	0	M	0	0	0	0	0	0	...	12.0	1.521424	28.969548	61	6.5	1	8/7/2012	B	1	-1
99997	99998	7/23/2012	1	M	0	0	1	0	0	0	...	12.0	1.025677	26.354919	61	6.9	1	7/27/2012	C	4	1
99998	99999	12/19/2012	0	M	0	0	0	0	0	0	...	16.0	1.035400	29.193462	59	5.6	1	12/23/2012	B	4	-1
99999	100000	3/6/2012	0	F	0	0	0	0	0	0	...	12.0	0.813004	25.175760	74	6.5	4	3/8/2012	B	2	-1

100000 rows × 29 columns

- Here is a list of the features involved.

In [115]:
```
df_features = pd.read_csv('C:\\Users\\javyr\\Documents\\GitHub\\r-server-hospital-length-of-stay\\Resources\\Data_Dictionary.csv')
df_features
```

Out[115]:

	Index	Data fields	Type	Descriptions
0	1	eid	Integer	Unique Id of the hospital admission
1	2	vdate	String	Visit date
2	3	rcount	Integer	Number of readmissions within last 180 days
3	4	gender	String	Gender of the patient\nM or F
4	5	dialysisrenalendstage	String	Flag for renal disease during encounter
5	6	asthma	String	Flag for asthma during encounter
6	7	irondef	String	Flag for iron deficiency during encounter
7	8	pneum	String	Flag for pneumonia during encounter
8	9	substancedependence	String	Flag for substance dependence during encounter
9	10	psychologicaldisordermajor	String	Flag for major psychological disorder during e...
10	11	depress	String	Flag for depression during encounter
11	12	psychother	String	Flag for other psychological disorder during e...
12	13	fibrosisandother	String	Flag for fibrosis during encounter
13	14	malnutrition	String	Flag for malnutrituion during encounter
14	15	hemo	Float	Flag for blood disorder during encounter
15	16	hematocritic	Float	Average hematocritic value during encounter (g...
16	17	neutrophils	Float	Average neutrophils value during encounter (c...
17	18	sodium	Float	Average sodium value during encounter (mmol/L)
18	19	glucose	Float	Average sodium value during encounter (mmol/L)
19	20	bloodureanitro	Float	Average blood urea nitrogen value during encou...
20	21	creatinine	Float	Average creatinine value during encounter (mg/dL)
21	22	bmi	Float	Average BMI during encounter (kg/m2)
22	23	pulse	Float	Average pulse during encounter (beats/m)
23	24	respiration	Float	Average respiration during encounter (breaths...
24	25	secondarydiagnosisnonicd9	Integer	Flag for whether a non ICD 9 formatted diagnos...
25	26	discharged	String	Date of discharge
26	27	facid	Integer	Facility ID at which the encounter occurred
27	28	lengthofstay	Integer	Length of stay for the encounter

- *Pre-processing*: Since the features 'vdate', 'gender', 'discharged' and 'facid' consisted of string values, they were excluded from the list of features for the analysis. In addition, these points don't even relate to the health of the patient. Lastly, 'rcount' was removed since it was used to create 'Readmissions'.
- *Data Source*: https://github.com/Microsoft/r-server-hospital-length-of-stay/tree/master/Data (https://github.com/Microsoft/r-server-hospital-length-of-stay/tree/master/Data)
- *Extra Source*: http://www.sciencedirect.com/science/article/pii/S1532046415000969 (http://www.sciencedirect.com/science/article/pii/S1532046415000969)

3. Methods

In [101]:
```
# Features attributes included in the analysis.
df_feature_names = ['eid', 'dialysisrenalendstage', 'asthma',
       'irondef', 'pneum', 'substancedependence', 'psychologicaldisordermajor',
       'depress', 'psychother', 'fibrosisandother', 'malnutrition', 'hemo',
       'hematocrit', 'neutrophils', 'sodium', 'glucose', 'bloodureanitro',
       'creatinine', 'bmi', 'pulse', 'respiration',
       'secondarydiagnosisnonicd9', 'lengthofstay']
# Separate the data from the target attributes.
X = df.loc[:,df_feature_names].values
y = df['Readmission'].values
```

In [102]:
```
# Used the 'train_test_split' function and set 'test_size' = 0.3 to randomly split 'X' and 'y' into separate training
# and test datasets, where 70 percent of the samples were assigned to 'X_train' and 'y_train', and 30 percent to
# 'X_test' and 'y_test', respectively.
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.30,random_state=0)
print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

```
(70000, 23) (30000, 23) (70000,) (30000,)
```

In [103]:
```
from sklearn.preprocessing import StandardScaler

# Scaling the data using the 'StandardScaler'.
stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(X_train)
X_test_std = stdsc.transform(X_test)
```

- Since this work deals with a binary classification problem, one of the most powerful approaches to tackle a problem such as this can be a logistic regression model.
- Logistic regression is used to obtain odds ratio in the presence of more than one explanatory variable. The procedure is quite similar to multiple linear regression, with the exception that the response variable is binomial. The result is the impact of each variable on the odds ratio of the observed event of interest.

In [117]:
```
#from the linear_model module of sklearn import the LogisticRegression class
from sklearn.linear_model import LogisticRegression

#lr is an object from the LogisticRegression class
lr = LogisticRegression()

#fitting the model to the labeled training set
#X_train: input data, y_train: target attribute (labels)
modele = lr.fit(X_train_std,y_train)

#the coefficients and the intercept
print(modele.coef_,modele.intercept_)
```

```
[[ 0.00192766 -0.15009026 -0.17327901 -0.25836592 -0.15103778 -0.25245198
  -0.61422537 -0.11266596 -0.19269591 -0.11242514 -0.12602399 -0.39091032
  -0.04083838 -0.11294494 -0.01100164 -0.01393363 -0.14508847  0.00313519
   0.00690782  0.00480912  0.00488537  0.01195433  2.57816064]] [-0.07136727]
```

In [111]:
```
#prediction on the test sample
y_pred = modele.predict(X_test_std)

#metrics - quantifying the quality of the prediction
from sklearn import metrics

#confusion matrix : comparing observed target values and predictions
cm = metrics.confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)

#accuracy rate
acc = metrics.accuracy_score(y_test,y_pred)
print("Accuracy Rate")
print(acc) # 0.819 = (14488 + 10100)/ (14488 + 1889 + 3523 + 10100)

#error rate
err = 1.0 - acc
print("Error Rate")
print(err) # 0.180 = 1.0 - 0.819
```

```
Confusion Matrix
[[14488  1889]
 [ 3523 10100]]
Accuracy Rate
0.8196
Error Rate
0.1804
```

- The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features.
- First, the estimator is trained on the initial set of features and weights are assigned to each one of them.
- Then, features whose absolute weights are the smallest are pruned from the current set features.
- That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

In [112]:
```
from sklearn.feature_selection import RFE

rfe = RFE(lr, 3)
fit = rfe.fit(X_train_std,y_train)

print ("Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), fit.ranking_), df_feature_names)))
```

```
Features sorted by their rank:
[(1, 'hemo'), (1, 'lengthofstay'), (1, 'psychologicaldisordermajor'), (2, 'irondef'), (3, 'psychother'), (4, 'substancedependence'), (5, 'bloodureanitro'), (6, 'asthma'), (7, 'pneum'), (8, 'dialysisrenalendstage'), (9, 'malnutrition'), (10, 'neutrophils'), (11,
 'depress'), (12, 'fibrosisandother'), (13, 'hematocrit'), (14, 'glucose'), (15, 'secondarydiagnosisnonicd9'), (16, 'sodium'), (17, 'bmi'), (18, 'respiration'), (19, 'pulse'), (20, 'creatinine'), (21, 'eid')]
```

- From the results of RFE, the 3 most important features are: 1) hemo - blood disorder 2) lengthofstay - length of stay for the encounter 3) psychologicaldisordermajor - major psychological disorder

In [113]:
```
#prediction on the test sample
y_pred = fit.predict(X_test_std)

#metrics - quantifying the quality of the prediction
from sklearn import metrics

#confusion matrix : comparing observed target values and predictions
cm = metrics.confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)

#accuracy rate
acc = metrics.accuracy_score(y_test,y_pred)
print("Accuracy Rate")
print(acc) # 0.808 = (14050 + 10177)/ (14050 + 2327 + 3446 + 10177)

#error rate
err = 1.0 - acc
print("Error Rate")
print(err) # 0.192 = 1.0 - 0.808
```

```
Confusion Matrix
[[14050  2327]
 [ 3446 10177]]
Accuracy Rate
0.807566666667
Error Rate
0.192433333333
```

- Based on the results, we can conclude that the logistic regression model can be a good model candidate to implement in problems such as this one in order to obtain high accuracy, which leads to a low error rate.